Report No. NAWCADWAR-95004-4.5

OPERATION COMPLEXITY FOR INTEGER OR RNS GAUSSIAN ELIMINATION

Peter R. Turner, Ph.D.
Mathematics Department
U.S. NAVAL ACADEMY
Annapolis, MD 21402

Barry Kirsch
Avionics Department
Engineering Division (Code 4.5.5.1)
NAVAL AIR WARFARE CENTER
AIRCRAFT DIVISION WARMINSTER
P.O. Box 5152
Warminster, PA 18974-0591

1 NOVEMBER 1994

FINAL REPORT

Approved for Public Release; Distribution is Unlimited.

Prepared for
OFFICE OF NAVAL RESEARCH ONR-313
800 N. Quincy St.
Arlington, VA 22217-5660

NOTICES

REPORT NUMBERING SYSTEM - The numbering of technical project reports issued by the Naval Air Warfare Center, Aircraft Division, Warminster is arranged for specific identification purposes. Each number consists of the Center acronym, the calendar year in which the number was assigned, the sequence number of the report within the specific calendar year, and the official 2-digit correspondence code of the Functional Department responsible for the report. For example: Report No. NAWCADWAR-95010-4.6 indicates the tenth Center report for the year 1995 and prepared by the Crew Systems Engineering Department. The numerical codes are as follows.

Code	Department
4.1	Systems Engineering Department
4.2	Cost Analysis Department
4.3	Air Vehicle Department
4.4	Propulsion and Power Department
4.5	Avionics Department
4.6	Crew Systems Engineering Department
4.10	Conc. Analy., Eval. and Plan (CAEP) Department

PRODUCT ENDORSEMENT - The discussion or instructions concerning commercial products herein do not constitute an endorsement by the Government nor do they convey or imply the license or right to use such products.

Reviewed By: Barry Kirsch **Date:** 2/16/95
Author/COTR

Reviewed By: _____ **Date:** 2/9/95
LEVEL III Manager

REPORT DOCUMENTATION PAGE

Form Approved
OMB No. 0704-0188

1. AGENCY USE ONLY *(Leave blank)*	2. REPORT DATE	3. REPORT TYPE AND DATES COVERED
	1 NOV 1994	FINAL

4. TITLE AND SUBTITLE

OPERATION COMPLEXITY FOR INTEGER OR RNS GAUSSIAN ELIMINATION

5. FUNDING NUMBERS

6. AUTHOR(S)

*PETER R. TURNER, PH.D. and **BARRY J. KIRSCH

7. PERFORMING ORGANIZATION NAME(S) AND ADDRESS(ES)

* Mathematics Dept.
U.S. NAVAL ACADEMY
Annapolis, MD 21402

** Avionics Department
Engineering Division (Code 4.5.5.1)
NAVAL AIR WARFARE
AIRCRAFT DIVISION WARMINSTER
P.O. Box 5152, Warminster, PA 18974-0591

8. PERFORMING ORGANIZATION REPORT NUMBER

NAWCADWAR-95004-4.5

9. SPONSORING / MONITORING AGENCY NAME(S) AND ADDRESS(ES)

OFFICE OF NAVAL RESEARCH
800 N. Quincy St.
Arlington, VA 22217

10. SPONSORING / MONITORING AGENCY REPORT NUMBER

11. SUPPLEMENTARY NOTES

12a. DISTRIBUTION / AVAILABILITY STATEMENT

APPROVED FOR PUBLIC RELEASE; DISTRIBUTION IS UNLIMITED.

12b. DISTRIBUTION CODE

13. ABSTRACT *(Maximum 200 words)*

This note addresses the question raised by Turner & Kirsch (1994) of the operation counts for the Gauss elimination solution of adaptive beamforming problems using integer arithmetic. Because the covariance matrix is positive definite hermitian, it follows that the multipliers cannot be precomputed and stored for each pair of rows. This has the effect of increasing the number of divisions from $O(N^2)$ to $O(N)^3$ which for any integer arithmetic (and, especially, RNS arithmetic) may prove to be an unacceptable cost. These results are extended to various degrees of parallelism in the integer or RNS processors and to the use of the L-CRT for scaling in a divisionless algorithm. Scaling using a fractional divider is also considered. The cost of RNS divisions is revisited in the light of newer division algorithms based on the work of Hitz and Kaltofen (1994). The relative cost of the divisions is substantially reduced rendering the RNS approach potentially practical for moderate-size problems.

14. SUBJECT TERMS

GAUSS ELIMINATION SOLUTION
RESIDUE NUMBER SYSTEM (RNS)

15. NUMBER OF PAGES

16. PRICE CODE

17. SECURITY CLASSIFICATION OF REPORT	18. SECURITY CLASSIFICATION OF THIS PAGE	19. SECURITY CLASSIFICATION OF ABSTRACT	20. LIMITATION OF ABSTRACT
UNCLASSIFIED	UNCLASSIFIED	UNCLASSIFIED	SAR

Operation Complexity for Integer or RNS Gaussian Elimination

PETER R TURNER AND BARRY J KIRSCH

Mathematics Department, US Naval Academy, Annapolis, MD 21402 and
Naval Air Warfare Center, Code 5051, Warminster, PA 18974

ABSTRACT. This note addresses the question raised by Turner & Kirsch (1994) of the operation counts for the Gauss elimination solution of adaptive beamforming problems using integer arithmetic. Because the covariance matrix is positive definite hermitian, it follows that the multipliers cannot be precomputed and stored for each pair of rows. This has the effect of increasing the number of divisions from $O(N^2)$ to $O(N^3)$ which for any integer arithmetic (and, especially, RNS arithmetic) may prove to be an unacceptable cost. These results are extended to various degrees of parallelism in the integer or RNS processors and to the use of the L-CRT for scaling in a divisionless algorithm. Scaling using a fractional divider is also considered. The cost of RNS divisions is revisited in the light of newer division algorithms based on the work of Hitz and Kaltofen (1994). The relative cost of the divisions is substantially reduced rendering the RNS approach potentially practical for moderate-size problems.

1. INTRODUCTION

This paper follows up on the observation made in [10] that the Gaussian elimination solution of the covariance matrix form of the adaptive beamforming problem using integer arithmetic requires a modification to the basic elimination algorithm so that the divisions are performed after double-length multiplications. The choice of Gaussian elimination for adaptive beamforming was discussed in [6] and [7]. The essential fact is simply that the covariance matrix is positive definite so that simple Gauss elimination is stable. For adaptive processing the covariance matrix is changing each time the system must be solved and so there is no advantage in storing the factors of the LU decomposition of the matrix. Gauss elimination is also more economical in terms of arithmetic operation counts than alternatives. In section 3 of [10] the conventional *ijk*-form of the forward elimination algorithm is described as:

for $i = 1$ to $N-1$
  for $j = i+1$ to N
    $m := a_{ji}/a_{ii}$
    $a_{ji} := 0$
    $b_j := b_j - mb_i$
    for $k = i+1$ to N
      $a_{jk} := a_{jk} - ma_{ik}$

The problem here is that for the adaptive beamforming problem the covariance matrix is positive definite Hermitian and so has its largest element on the diagonal. On at least one occasion (using integer arithmetic) the multiplier m will therefore round to 0 leading to erroneous solutions. The simple way to avoid this is to perform the multiplications using a double-length accumulator first and then to divide these long products by the appropriate diagonal element. The dynamic range and precision analysis of [10] was based on the use of this strategy.

In this paper we address the questions raised by this in terms of arithmetic operation counts and therefore timing for both conventional (binary) integer arithmetic and for residue number system, RNS, arithmetic. In Section 2 we look at the division operations for forward elimination and estimate their cost in RNS arithmetic in terms of a base unit of an 8-bit arithmetic operation such as would be performed for each modulus in RNS. The primary conclusion from this section is that these divisions achieved by using the Chinese Remainder Theorem, CRT, would make the process too expensive.

Section 3 deals with what may be regarded as the opposite extreme approach for integer Gaussian elimination - the divisionless algorithm in which all the elimination steps are based on "cross-multiplication" between the rows. We see that only the very smallest and simplest of adaptive beamforming problems are amenable to this approach due to the rapid growth of the required dynamic range.

Clearly, some compromise between these extremes is necessary if integer, and especially RNS, Gauss elimination is to be a practical tool for these problems. In Section 4 we study the possibility of computing and storing the multipliers as fractions and then scaling the "elimination row" appropriately. The idea being to reduce the number of divisions that are to be performed back to $O(N^2)$. Unfortunately the cost in terms of dynamic range growth remains too great for problems with $N > 4$.

In [6] and [7] we considered the use of a modified divisionless algorithm incorporating some scaling which could be performed in RNS using the L-CRT which is a modification of the CRT with a built-in scale factor and a consequently shorter binary accumulator. This approach is analysed in Section 5. The principal conclusion there is that some 70% of the cost of the divisions can be saved making the RNS solution feasible for small beamforming problems. The proportional saving decreases to about 35% for larger dimensional problems showing that this approach is clearly superior to using the CRT to perform the divisions. It still appears that the overall cost would become excessive as the dimension increases.

An alternative to scaling would be to use one of the more recent RNS division algorithms such as that of Hitz and Kaltofen [5]. This algorithm makes use of the RNS equivalent of a double length accumulator to perform the divisions by using an extended RNS basis. Its efficiency relies heavily on having sufficient parallelism in and among the RNS processors. The use of, and improvements to, this algorithm within

our Gauss elimination scheme are the subject of Section 6. The improvements which are likely to be obtained using this division algorithm suggests that beamforming problems of considerably greater dimension may be amenable to solution using RNS arithmetic.

2. Division count for forward elimination

The modified form of the algorithm, still written in the ijk–form would be

for $i = 1$ to $N-1$
  for $j = i+1$ to N
    $b_j := b_j - (b_i a_{ji})/a_{ii}$
    for $k = i+1$ to N
      $a_{jk} := a_{jk} - (a_{ik}a_{ji})/a_{ii}$
    $a_{ji} := 0$

We see that the number of divisions is now exactly the same as the number of multiplications, namely $N(N^2-1)/3$ rather than the $N(N-1)/2$ which are required for the original form. (See [1], Chapter 6, for example.) Fast VSLI designs for hardware integer division using a "double-length" accumulator use essentially the same architectural components as double-length multipliers and so we anticipate a significant time-penalty for conventional binary integer arithmetic.

Note that the back substitution phase of the solution is unaffected by the need to perform the divisions last as this is the natural arrangement for that stage of the Gauss elimination process.

For problems of the same dimensions as those considered in [10] where the number of antenna elements (and therefore the dimension of the linear system) N varied from 4 to 8 to 16 the number of divisions needed in the forward elimination phase are given in Table 2.1 below.

TABLE 2.1
Numbers of divisions required by original algorithm
and using "long-accumulator" version

# antennas N	$N(N-1)/2$	$N(N^2-1)/3$
4	6	20
8	28	168
16	120	1360

For ordinary binary integer arithmetic the time-penalty associated with this increased number of divisions is likely to be significant. For a special arithmetic such as RNS in which division is not easily achieved it is much more troublesome.

Division cannot be performed entirely within the RNS system but requires a conversion to standard binary form via the Chinese Remainder Theorem. Multiplication (even complex multiplication) is readily achieved in RNS and benefits from the natural parallelism inherent in the system which permits all RNS arithmetic to be per-

formed using short wordlengths. Indeed complex multiplication can be performed in the Galois-enhanced quadratic RNS (GEQRNS) system in a time comparable to two RNS real additions. This is achieved by using representations of both the complex integers z and $\overline{z}$ relative to Gaussian prime base moduli and then using number-theoretic logarithms of these quantities to perform the multiplication. For a system using an effective binary wordlength of say 128 bits but with individual moduli of say 8 bits, it follows that such a complex multiplication is equivalent to just two 8-bit additions if the processors for all the different moduli can be operated simultaneously.

Alternatively, multiplications for both the complex factors and their conjugates could be performed using two modular multiplication operations. Even in the simple implementation of the RNS system for complex arithmetic, CRNS, using RNS representations of both the real and imaginary parts, complex multiplication is performed using four real multiply-accumulate operations. This complex multiplication will only take a time comparable with four real 8-bit multiplications again assuming that the different moduli are processed in parallel.

A corresponding division however first requires conversion to a 128-bit binary integer format and then two 128-bit integer divisions - one for each component of the complex numerator. (The divisor is always real in the case of a positive definite Hermitian matrix.) The time-penalty for these divisions in RNS arithmetic is therefore quite severe.

First consider the CRT conversion from sixteen 8-bit moduli to a single 128-bit binary integer. Denote the moduli by $m_1, m_2, ..., m_{16}$ and their product by M. Further we write $\widehat{m_i} = M/m_i$. Here we assume that $m_1, m_2, ..., m_{16} < 256$ so that each is representable in 8 bits and the product $M < (2^8)^{16} = 2^{128}$. Then any nonnegative integer $a < M$ is uniquely represented by its residues $a_1, a_2, ..., a_{16}$ relative to the various moduli. That is

$$a_i \equiv a \bmod m_i =: \langle a \rangle_{m_i}$$

Then a is given by the CRT by

$$a = \left\langle \sum_{i=1}^{16} \widehat{m_i} \left\langle a_i \widehat{m_i}^{-1} \right\rangle_{m_i} \right\rangle_M$$

Assuming that the quantities $\left\langle \widehat{m_i}^{-1} \right\rangle_{m_i}$ are available as stored constants, the inner multiplication is just an 8-bit modular multiplication and each of these can be performed simultaneously. These are followed by multiplications of factors with 120-bit and 8-bit wordlengths and accumulation of the products. This multiplication could be achieved using 8 steps of a double-length multiplier for 64-bit words. (A full double-length 64-bit multiplier would need 64 such steps.) The modular part of this operation is insignificant, the principal cost is equivalent to $16 \times 8 + 15 = 143$

128-bit accumulations. This must be performed for both real and imaginary parts of the numerator and the real denominator.

The division can then be performed using this same accumulator using a further 64 steps. (Here we have taken account of the fact that if all products are to be accommodated in the 128-bit equivalent binary system then the individual elements of the matrix must have real and imaginary components representable in 64 bits.) Finally there is the conversion back to RNS of the quotient which again will be negligible compared to the costs already accounted for. The division operation in this case is therefore equivalent to some $3(143)+64 = 493$ accumulations in a 128-bit accumulator. This estimated cost has completely ignored any overhead.

Even if we assume the practicality of a 128-bit Carry Look-Ahead adder (which is certainly still in the future) so that we may take accumulator times to be approximately proportional to the $\log_2$(wordlength) then these 128-bit accumulations will take about 3 times as long as the modular 8-bit operations. That is each division will take something around 1500 8-bit modular operation times. Depending on the details of the complex RNS arithmetic implementation this is equivalent to saying that a division is equivalent to somewhere between 400 and 800 multiplications.

3. Problem Size and the Divisionless Algorithm

The cost of divisions whether in conventional integer arithmetic or using RNS indicated by the analysis of the previous section behooves us to consider the divisionless form of the Gauss elimination algorithm which was discussed in [6] and [7]. The corresponding ijk-form of this algorithm is

for $i = 1$ to $N-1$
  for $j = i+1$ to N
    $b_j := a_{ii}b_j - a_{ji}b_i$
    for $k = i+1$ to N
      $a_{jk} := a_{ii}a_{jk} - a_{ji}a_{ik}$
    $a_{ji} := 0$

The major difficulty with this form of the algorithm is of course that the dynamic range (or, equivalently, wordlength) required grows rapidly as the elimination proceeds. The rate of this growth and possible schemes for dealing with it are discussed in [6] and [7] and the additional requirements consequent on the precision demanded in the final weights are discussed in [10]. Our primary purpose in this section is to consider the size of problem and the input data precision which can be accommodated using the divisionless algorithm without any scaling and assuming some limit on the largest wordlength available.

Denote by $W+1$ the maximum integer wordlength. For simplicity and for consistency with the RNS system, we shall assume a sign-magnitude representation so that the corresponding dynamic range consists of integers in the interval $\left(-2^W, 2^W\right)$. We

shall denote the dimension of the system by N and the initial dynamic range of the real and imaginary parts of the matrix and right-hand side by $(-2^w, 2^w)$ corresponding to an initial wordlength of $1+w$ bits. At each stage of the elimination, the complex cross-multiplication and subtraction has the effect of approximately doubling the required wordlength. Specifically in the first stage of the elimination the *symmetric* wordlength w increases to $2w+2$. By the time the k-th stage has been completed the remaining elements of row $k+1$ and subsequent rows require a symmetric wordlength of $2^k w + 2(2^k - 1)$.

For example, at the end of the forward elimination for a 4×4 matrix the symmetric wordlengths of the four rows would be w, $2w+2$, $4w+6$, and $8w+14$.

Next consider the back substitution phase of the solution which is again to be performed in divisionless form. Again the basic algorithm is described in [6] using cross-multiplication in place of division. Also the final division is avoided by forcing the resulting diagonal elements to be identical and simply scaling (or reading off the appropriate number of initial bits from) the final right-hand side. (Note that this is permissible since any constant multiple of the true weights is acceptable for our beamforming problem.) This phase of the algorithm is described as follows:

for $i = N$ to 1
  for $j = 1$ to $i - 1$
    $b_j := a_{ii} b_j - a_{ji} b_i$
  for $j = i + 1$ to N
    $b_j := a_{ii} b_j$

The final solution is then just the appropriately scaled final right-hand side vector. The corresponding cross-multiplications for the left-hand side are not needed explicitly. The main advantage of this is that the dynamic range growth occurs on the right-hand side only. This in turn reduces that growth since the matrix elements used remain in their initial range of this phase.

Consider, now, the dynamic range growth resulting from this phase of the solution process. It will be convenient to denote the symmetric wordlengths of the various rows at the beginning of the back substitution by w_i. From the earlier analysis, we have the initial values

$$w_{i+1} = 2^i w + 2(2^i - 1)$$

for each $i = 0, 1, ..., N-1$. On the first step, every element on the right-hand side, except for the final one, undergoes a cross-multiplication with elements of the final row. The resulting dynamic range for these modified elements of the right-hand side thus become:

$$w_i + w_N + 2 \qquad (i = 1, 2, ..., N-1)$$

while the final row remains unchanged.

At the next step, the first $N-2$ rows are similarly modified by cross-multiplying with elements of the $(N-1)$-th row of the matrix while the final component of the right-hand side vector is simply multiplied by $a_{N-1,N-1}$ which, since this multiplier is real, simply increases the required wordlength by w_{N-1}. The symmetric wordlengths are now

$$\begin{array}{ll} w_i + w_{N-1} + w_N + 4 & (i = 1, 2, ..., N-2) \\ w_{N-1} + w_N + 2 & \\ w_{N-1} + w_N & \end{array}$$

Continuing in this way, we obtain symmetric wordlengths at the end of the back substitution given by

$$2(N-i) + \sum_{j=1}^{N} w_j \qquad (i = 1, 2, ..., N)$$

However, since these are all similarly scaled multiples of the true solution, it follows by symmetry that each of these is representable in the same wordlength. It follows that the symmetric wordlength needed for this right-hand side vector at the conclusion of the back substitution is

$$\widehat{W} = \sum_{j=1}^{N} w_j = (2^N - 1)w + 2\left(2^N - N - 1\right)$$

In essence, we see that the back substitution can result in (almost) doubling the largest wordlength required in the forward elimination.

The first and most striking conclusion to be drawn is that for any $N \geq 7$, $\widehat{W} \geq 255$ and so such problems cannot be solved using an entirely divisionless algorithm with accumulators of 256 or fewer bits. In fact for the adaptive beamforming problem, if the data "snapshots" are *quantized* to q_X bits as in [10] then $w \geq 2q_X + 1$ so that even with the minimal requirement $q_X = 1$ (which is sufficient only to distinguish between "positive", "zero" and "negative") then $w \geq 3$ and so problems with $N \geq 6$ are not solvable in this wordlength. Of the problem sizes considered in [10] we can therefore only hope to solve problems of dimension $N = 4$ using this divisionless algorithm without any scaling.

For $N = 4$, the above equation becomes $\widehat{W} = 15w + 22$. For maximal accumulators of 128 and 256 bits, that is $W = 127$ and $W = 255$, we therefore require $15w \leq 105$ or $15w \leq 233$ respectively. These correspond to maximal values $w = 7$ and $w = 15$ or $q_X = 3$ and $q_X = 7$ respectively. According to the conclusions of [10] using the weight quantization requirements obtained by Nitzberg [9] this is probably just sufficient for a maximum jammer SNR of 20dB.

There is of course the possibility that the initial computation of the covariance matrix and cross-correlation vector is performed separately and then rounded to the 7 or 15-bit wordlengths. Even this would only allow the most significant part of each component of the scalar products to be retained for data quantization of these same 7 or 15 bit precisions. This does not affect the scale of problem solvable in 128 bits. With $W = 255$, and still with $N = 4$ this scaling of the initial matrix and right-hand side does permit solution of problems with jammer SNR's up to about 40dB though there is some inevitable loss of precision in the final weights.

The obvious conclusion from this section is that some scaling or some divisions are necessary if adaptive beamforming problems are to be solved by Gauss elimination using integer arithmetic. We address these questions further in the subsequent sections.

4. Scaling by use of a fractional divider

In this short section we investigate the possibility of using the standard Gauss elimination algorithm with integer arithmetic by computing the multipliers using a fractional divider and then scaling the elimination row and the multiplier to account for (at least some of) the fraction bits. More specifically, the original form of the algorithm given in Section 1 is used except that the multipliers m are computed as fractions to the full precision of the current wordlengths. This fraction is then scaled to an integer by multiplication by the appropriate power of 2 and each element of row j is similarly scaled before the elimination proceeds.

There are two principal effects of this. First the number of divisions is reduced back to the level of the standard *floating-point* algorithm - that is $N(N-1)/2$ divisions are now required in the forward elimination phase. There is some additional overhead. Counting the number of fractional bits is essentially automatic as part of the division algorithm since the numerator and denominator must be appropriately aligned at the outset. However this scale factor, s, and the "integerized" multiplier must be converted back to RNS and then the elimination arithmetic performed as

$b_j := sb_j - mb_i$
for $k = i + 1$ to N
$a_{jk} := sa_{jk} - ma_{ik}$

These additional operations are entirely within RNS however and so their cost relative to even the single division is almost negligible.

The second important aspect to consider is the effect of this scaling on the dynamic range requirements of the algorithm. Clearly at any stage there is the possibility that the true multiplier can be very small and so the scale factor can potentially be equivalent to multiplying the rows by the largest quantity currently representable. This means we have to permit the dynamic range to grow at the same rate as for the divisionless algorithm. In which case, of course, we may as well use the divisionless

algorithm and avoid the divisions altogether.

There is some potential for a compromise. If the fractional multipliers are computed to somewhat reduced accuracy, then the advantage of reducing the number of divisions is retained but the scale factors will be correspondingly smaller - and therefore, so will the dynamic range growth. This of course introduces rounding errors and therefore errors in the weights obtained. The effect of these rounding errors on the weights is expected to be bounded by multiplication by the condition number of the matrix which can be very large. For example, using the analysis of Compton [2], the condition number of the covariance matrix is equal to the jammer SNR so that for the cases considered in [10] the condition number varies between 100, 1000, 10000 and 100000 or approximately 2^7, 2^{10}, 2^{13}, and 2^{16}.

This implies that the effect of the roundoff errors introduced by using a truncated fraction may be expected to be magnified by the loss of a further 7, 10, 13 or 16 bits respectively. In practical experimentation, Monzingo and Miller [8] found a much more optimistic trend. With an eigenvalue spread of 40dB, or a condition number of around 2^{13}, and using only a short computational wordlength they found a degradation of only 2dB in the weights, or a loss of only about 1 bit of precision. However we should observe that this is an average and details of the algorithm used or the test methodology are not provided. Also if we are to compare those findings with the eigenvalue analysis, it would be necessary to have statistics relating to the worst-case loss of precision. If reliable bounds are to be used we are therefore obliged to use the more pessimistic eigenvalue results.

Now the weight quantizations required for the various problem sizes considered in [10] were tabulated from Nitzberg's results. They are reproduced as Table 4.1 below.

TABLE 4.1

Weight quantization for a 3dB degradation as a function of N and the jammer SNR from [9]

N	20 dB	30 dB	40 dB	50 dB
4	6	9	13	16
8	7	10	14	17
16	8	11	15	18

For each of these columns, using the eigenvalue analysis there is a potential loss of precision which needs to be added to the computational weight quantization wordlength. These effective wordlengths are summarized in Table 4.2.

These wordlengths represent the minimum fraction precision which could be used in this approach of reducing the division count by computing fractional multipliers

and then scaling. They also therefore represent the range of potential scale factors - or equivalently, the potential dynamic range growth at each stage of the elimination. In all cases these wordlengths are smaller than the initial wordlengths required for storage of the covariance matrix and cross-correlation vector for the data quantizations obtained in [10].

TABLE 4.2

Weight quantization wordlengths for a 3dB degradation as a function of N and the jammer SNR from [9] allowing for loss of precision due to roundoff errors using the condition numbers of [2]

N	20 dB	30 dB	40 dB	50 dB
4	13	19	26	32
8	14	20	27	33
16	15	21	28	34

Using the same notation w_i as in the previous section for the symmetric wordlengths at the end of the elimination phase, we now have

$$w_i = w + (i-1)Q$$

where, as before, w represents the symmetric wordlength for the initial matrix and Q the weight quantization wordlength from Table 4.2.

If the divisionless back substitution is to be used to complete the solution, we find that

$$\widehat{W} = \sum_{i=i}^{N} w_i = Nw + \frac{N(N-1)Q}{2}$$

It is clear that for $N \geq 8$ we have $\widehat{W} \geq 255$ which was the larger of the two values of W we considered in Section 3. For $N = 4$, the values of w, Q from [10] and Table 4.2 above are 17, 13 for 20dB; 23, 19 for 30dB; 31, 26 for 40dB. (The values for w are obtained from the values of $2q_X + 1$ in Table 5 of [10].) The corresponding values of $\widehat{W}$ are 146, 206 and 280 so that we can expect to obtain solutions for four antenna problems with a jammer SNR no greater than about 35dB.

We should note here that if the loss of precision observed by Monzingo and Miller [8] is to be trusted in practice then the growth factors are much reduced. Specifically the wordlengths in Table 4.2 would be replaced by entries which are just one or two greater than those of Table 4.1. The observation that $\widehat{W} \geq 255$ for $N \geq 8$ remains valid. For $N = 4$, the values of w, Q for a 50dB jammer are 37 and 18 so that

$\widehat{W} = 256$ and given that maximal growth will not happen at every single stage we can be confident of solving such a problem with a 256-bit accumulator limit. It is reasonable to expect therefore that a four antenna adaptive problem can be solved using a fractional divider in the manner described but that that is the limit of the usefulness of this approach.

Thus scaling using fractional division helps - but not enough.

5. Using the L-CRT for scaling in the divisionless algorithm

In this section we consider the special form of the divisionless algorithm discussed in [6] and [7] in which the range growth was controlled by various scaling operations which could be performed using the L-CRT [4] which is a modification of the regular CRT in which a full-length accumulator is unnecessary. The basic idea is that each element is scaled by a real number V into a range M' which is a power of 2. Thus

$$M = VM'$$

and the L-CRT calculation is superficially similar to the regular CRT:

$$a' = \left\langle \sum_{i=1}^{L} \left\lfloor \frac{\widehat{m_i}}{V} \left\langle \widehat{m_i}^{-1} a_i \right\rangle_{m_i} \right\rfloor \right\rangle_{M'}$$

where L is the dimension of the RNS basis. This scaling process does, of course, introduce errors into the process but these are generally small since the error in $\lfloor \cdot \rfloor$ is bounded by 1 so that

$$\left| \frac{a}{V} - a' \right| < L \ll M$$

The multipliers $\widehat{m_i}/V$ could be stored to sufficient accuracy to reduce this error to within any desired bound if some fractional bits are permitted in the $\log M'$-bit accumulator. The big difference between this and the standard CRT lies in the fact that a shorter binary accumulator is used and the mod M' operation is automatic in such an accumulator.

The potential advantages offered by this for our divisionless algorithm are that these scaling operations are cheaper than full divisions and fewer of them are needed anyway. We should observe however that the detailed algorithm of [7] was only constructed for the $N = 4$ case. Because of the range growth that is allowed there, for a larger system it would be necessary to scale all subsequent stages of the elimination to the same range and to modify the back substitution to take account of the additional steps too. The alternative would be to add yet more moduli to the RNS basis but this would eventually necessitate a very large binary accumulator for the final CRT

conversions as well as a final mod operation relative to the resulting very large M value.

In the remainder of this section we consider the savings that can be obtained using this scaling. First we summarize the scaling operations that are needed during the process described in [7]. All elements of the initial matrix are scaled to a precision of ± 15 bits. (By this we mean a symmetric wordlength of 15 bits, or 15 bits plus a sign.) We may assume this initial scaling is done in the data channels before we begin RNS operations. This is adequate for the first stage of the elimination phase. The resulting 3×3 matrix and right-hand side must then be scaled to ± 27 bits so that a total of 12 L-CRT's using a 28-bit accumulator are needed. (The details of the scale factors are included in [6] and [7].)

Further scaling is needed for the back substitution phase: 2 elements are scaled using a 56-bit accumulator and then 4 using a maximal length of 80 bits and 4 more with a 96-bit accumulator. There are therefore a total of 22 quantities which must be scaled using L-CRT operations of varying sizes and using RNS bases of different dimensions. In estimating the timing of such operations we shall again ignore the modular multiplications at the innermost level of the L-CRT. Also we shall assume that accumulator lengths are in fact eight (or more) bits longer than those indicated above to allow for fractional bits in the accumulation of the sum. Since we assume all moduli are representable in eight bits this is sufficient to obtain the accuracy cited above. We shall also make the comparison with performing the divisions fair by assuming the availability of carry look-ahead adders of whatever lengths are needed. Again, we note that this is not a realistic assumption but was the basis for the pessimistic figures obtained in Section 2 for performing the divisions using the CRT and our comparison here is with those figures.

The scaling outlined above for the forward elimination involves 12 elements each represented using five moduli and an accumulator length of at least 36 bits. The terms being accumulated are therefore products of 8-bit and, say, 40-bit quantities with the final result being chopped to 28 bits. The multiplications require 8 steps of a 24-bit double-length accumulator for each modulus and with the final accumulation of the sum this yields a total of some 44 (that is, $5 \times 8 + 4$) cycles of a 48-bit accumulator. Such an accumulator, according to our assumptions, will be a factor of 2 slower than an 8-bit unit. There are 12 elements, nine of which are complex, to be scaled in this way so that this scaling is in total equivalent to approximately 1850 8-bit accumulator operations. Observe that this is in the range of a *single* CRT-based division.

The various scaling operations for the back substitution phase can be similarly analysed. In all cases there are 16 moduli, and a 128-bit accumulator is sufficient for all the operations. As with the divisions then 143 such accumulations are needed for each real or imaginary part to be scaled. There are 10 such quantities in all, and only one is real. This yields a total equivalent to $143 \times 3 \times 19$ or about 8150 8-bit

accumulations.

This in turn makes the total cost of the scaling operations for the complete solution process close to 10000 such 8-bit operations. For this same 4×4 system using CRT-based divisions as in Section 2, 24 such divisions are needed. The total cost of the scaling operations is very close to that of seven divisions and so the saving due to scaling by the L-CRT and the "divisionless" algorithm is approximately 70% of the cost of the divisions.

Using the same RNS bases as for the above analysis but extending to larger dimensional problems, we see that the forward elimination phase will use $N(N-1)$ scaling operations for the 5-dimensional basis and will then also need a further

$$(N-2)(N-3)+(N-3)(N-4)+\cdots+6=\frac{(N-1)(N-2)(N-3)}{3}-2$$

scalings with a 16-dimensional basis. The back substitution uses another $2+N(N-2)$ of these. The dominant part of this scaling cost therefore results from a total of $(N-2)(N^2-N+3)/3$ 16-dimensional L-CRT's for each of the real and imaginary parts. As in the analysis of the CRT each of these 16-dimensional operations is equivalent to 143 8-bit operations. With the same time-factor of 3 between the large accumulator and the 8-bit operations, we end up with $O(285N^3)$. From Table 2.1 the number of divisions in the integer form of Gauss elimination would be, including the back substitution phase, $N(N^2+2)/3$ with a total cost of $O(500N^3)$ so that the saving from using the L-CRT to perform the scaling decreases to around 35% as N increases. Unfortunately it remains $O(N^3)$ in the expensive calculation.

Scaling using the L-CRT is clearly beneficial and is probably sufficient to at least allow small dimensional problems to be solved using the scaled divisionless algorithm of [6] and [7].

6. Using improved RNS division algorithms

In this section we consider the use of more recent RNS-based division algorithms. In particular we study the use of an improved version of the algorithm of Hitz and Kaltofen [5] which can be used to obtain both the integer division and the remainder resulting from division of two integers each represented in an extended RNS which plays the role similar to that of a double length accumulator in regular binary arithmetic. No CRT conversions or their equivalent are required by this algorithm. It does however use several RNS base-extension operations. These however can be performed entirely within the RNS processors - provided enough such processors are available.

This section begins with a brief summary of the Hitz and Kaltofen algorithm which has at its heart a pseudo-reciprocation step. We then describe an improvement to this reciprocation which alleviates the need for the magnitude comparison which is a necessary correction step in the original algorithm. Even though these comparisons

can be achieved in the extended RNS being used they do entail further base-extensions and are therefore relatively expensive. We take further advantage of this to improve the performance of the algorithm by speeding up its slowest part by a factor close to two. We conclude this section with a discussion of the savings that can be achieved by use of this algorithm in the context of Gauss elimination for adaptive beamforming.

6.1. The division algorithm of Hitz and Kaltofen. The basic idea here is that a double length RNS representation is used in the following manner. Let $p_1, p_2, ..., p_N, p_{N+1}, ..., p_{2N}$ be a set of prime numbers. (They only need to be relatively prime for this purpose but we shall need Gaussian primes for our algorithm in order to perform complex RNS arithmetic efficiently.) Assume they are ordered so that $p_i < p_{N+i}$ for each $i = 1, 2, ..., N$. In [5] slightly stronger order assumptions are made but the additional assumptions play no role; indeed even this assumption can be weakened as we will see in discussing the use of this algorithm later.

Let

$$M = \prod_{i=1}^{N} p_i \ , \qquad \overline{M} = \prod_{i=1}^{N} p_{N+i}$$

so that M represents the dynamic range of the *base RNS* and $M\overline{M}$ represents the range of the *extended RNS*. Some important (though immediately apparent) observations are made in [5]:

M, $\overline{M}$ are relatively prime and $M < \overline{M}$.

It follows that $M^{-1} \bmod \overline{M}$ exists. Also the first N moduli in the extended representation of an extended RNS number represent the remainder $\bmod \overline{M}$.

It also follows that multiplication of two numbers represented in the base RNS (that is, two numbers smaller than M) cannot overflow the extended RNS range.

The division algorithm then consists of two stages in order to obtain the integer quotient $\lfloor X/Y \rfloor$ and remainder $X \bmod Y$ where X, Y are positive integers in the base RNS range. In the first, the "reciprocal" $\lfloor M/Y \rfloor$ of Y is obtained using a modification of Newton's method. This is then used to generated the desired quotient and remainder.

The reciprocation algorithm is described in [5] as follows.

Algorithm RECIP

Input: Y
Output: $\lfloor M/Y \rfloor$
begin
$Z_1 \leftarrow 0$
$Z_2 \leftarrow 2$
while $Z_1 \neq Z_2$ **do**

$Z_1 \leftarrow Z_2$
$Z_2 \leftarrow \lfloor Z_1 * (2M - Y * Z_1) / M \rfloor$
if $M - Y * Z_2 < Y$ **then return** Z_2 **else return** $Z_2 + 1$
end.

The need for, and correctness of, the correction step is established in [5]. The iteration in the loop is just Newton's method in an "integerized" form. The division by M, equality testing and comparison are all achievable using the extended RNS basis in ways which are also detailed in [5]. We shall discuss later only those details which are needed. Once this M-reciprocation has been completed the rest of the division is straightforward. This algorithm is also given in [5]:

Algorithm DIVREM

Input: X, Y
Output: $\lfloor X/Y \rfloor$ and $X \bmod Y$
begin
$Q \leftarrow \lfloor X * \mathrm{RECIP}(Y) / M \rfloor$
$R \leftarrow X - Q * Y$
if $R < Y$ **then return** $Q,\ R$ **else return** $Q + 1,\ R - Y$
end.

The comparison and division by M are performed as in the RECIP algorithm.

6.2. Improved reciprocation. It is apparent that the critical part of this division algorithm is RECIP and that any savings which can be achieved there are worthwhile. In this subsection we consider two sources of improvement. The first is the elimination of the correction step by using the ceiling rather than the floor function in the iteration. The convergence analysis must be appropriately modified - or simplified - to take account of this. This removes the need for the final comparison which is implemented in extended RNS by using two base extension operations.

In [5], the number of steps of the Newton iteration is also discussed. For convenience of the analysis this is separated into two parts which can be thought of as a first stage of order $O(\log M)$ which achieves an iterate with a relative error of less than 75% and a second $O(\log \log M)$ which is the usual quadratic convergence behavior of Newton's method. In our setting, the first of these constitutes a potentially prohibitive number of iterations to achieve a suitably good estimate that the quadratic convergence takes over. A second consequence of the use of the ceiling function in the iteration is that it becomes easy to recognize when the iterates are well-removed from the desired reciprocal - and to make appropriate adjustments for this. The effect is that the number of iterations used in this first phase is reduced by an asymptotic factor of 2. This modified algorithm is described and analysed in detail in [11]. Results

are included there which illustrate clearly that substantial savings in the number of iterations required are indeed achieved by this version of the algorithm.

The modified algorithm is based on the reciprocation algorithm CRECIP which is given in [11] as

Algorithm CRECIP (The accelerated Ceiling version of RECIP)

Input: Y
Output: $\lceil M/Y \rceil$
begin
$Z_1 \leftarrow 0$
$Z_2 \leftarrow 2$
while $Z_1 \neq Z_2$ and $Z_1 - 1 \neq Z_2$ **do**
$Z_1 \leftarrow Z_2$
$Z_2 \leftarrow \lceil Z_1 * (2M - Y * Z_1) / M \rceil$
$= 2Z_1 - \left\lfloor \frac{YZ_1^2}{M} \right\rfloor$
if $Z_2 = 2Z_1$ then $Z_2 = \frac{3}{2}Z_1^2$
return Z_1
end.

6.3. Using RNS division in Gauss elimination. If the RNS division algorithm described above is used within our Gauss elimination algorithm, the dynamic range growth ceases to be a problem since there is then no growth [12] for a positive definite Hermitian system. This growth result has another important consequence for the use of the RNS division algorithm. Because of this and the effective double-length accumulator provided by the extended RNS system, we know that all our results are representable in the base RNS and therefore that this division algorithm can be used without any need for further scaling operations.

The great cost of the additional divisions referred to in Section 2 is greatly alleviated too. Because the divisor is the same for every operation in each step of the elimination, there is only one call to the CRECIP procedure per row. That is there are just $N-1$ such reciprocation operations in the entire forward elimination process.

Note too that the divisions that are needed for the back substitution operations have the same divisors and so, provided the reciprocals are stored from the forward elimination, no further calls to CRECIP are needed there - except for the reciprocation of the final value of a_{NN} which has not been computed during the elimination phase.

The various division operations can then be completed with just a modular multiplication and the division by M which requires a further base-extension. The MRS conversion discussed in [5] is expensive in hardware and is probably impracticable for our situation. (It requires approximately $[L(L+1)/2] \log M$ RNS processors which

for the wordlengths discussed in [10] could be in the tens of thousands.) However, once the operation CRECIP has been completed the use of the MRS conversion for completing the divisions can be pipelined so that the overall time-delay is minimized. This idea is discussed in Section 4.3 of [6] using a recurrence based on the algorithm of Gregory and Matula [3].

Each step of the recurrence requires modular arithmetic using a separate modulus in the basis. If several quantities are to be converted to MRS form, it follows that while the first number is being processed in modulus m_i the second can be processed simultaneously in the m_{i-1} processor and so on with the i-th being processed in the m_1 processor. In our setting the dimension of the linear system and the number of RNS basis elements will be of similar magnitude and so the largest time-delay would approximately correspond to twice the time for a single MRS conversion. The generation of the new moduli for the extended RNS basis can be simultaneous for each (extended) basis element and so adds just one further RNS operation time. The complete time for this operation is therefore of the order of $2L$ RNS operation times for all elements which are to be "extended" at any one stage of the process.

From the results in [11], it appears that the number of iterations required for the CRECIP algorithm is always likely to be bounded by $\log(M/Y)$ and usually would be significantly less than this. The analysis of required wordlengths for Gauss elimination using integer divisions suggests that the divisors Y are likely to be large (since the maximal element of each successive square submatrix lies on the diagonal) and therefore that a typical value to be expected for the reciprocals may be bounded by around 2^{20}. The timing estimates below are based initially on an average of 20 iterations being required for CRECIP which probably corresponds to a mean value for the reciprocal of around 2^{25} or more.

Each iteration requires 3 modular operations and a base extension which we have estimated at around $2L$ operations. Only N such reciprocals need be formed and so taking $L = 16$ for the dimension of the base RNS scheme the 20 iterations amount to a total of some $700N$ modular operations. Completing the divisions then involves just one more modular operation plus a base-extension - or maybe 33 further modular operations. Even taking no account of any pipelining in this phase, we see that the $O(N^3)$ part of the process becomes a mere $11N^3$ modular operations. (Refer to Table 2.1.) The total cost of the expensive divisions for the whole process is thus reduced by this algorithm to approximately $11N^3 + 700N$ modular operations which must be compared with approximately $500N^3$ for the CRT-based divisions or $285N^3$ for the L-CRT with scaling approach of Section 5.

In Table 6.1 these modular operation counts are compared for $N = 4, 8, 16$ using the 20 iterations per CRECIP mentioned above and with 40 and 60 iterations also to show that even if this estimate is very low the potential saving is great.

TABLE 6.1

Equivalent modular operation counts

N	CRT divisions	L-CRT scaling	RNS divisions		
			20 its	40 its	60 its
4	32 K	18 K	3.5 K	6.3 K	9.1 K
8	256 K	146 K	11 K	17 K	22 K
16	2 M	1.2 M	56 K	67 K	78 K

It is apparent that the saving resulting from the improved RNS division are great. The expected times for problems with $N = 16$ are of similar order of magnitude to those for $N = 4$ using the CRT for the divisions or even using the L-CRT and scaling. Thus our earlier conclusion that problems of dimension 4 were about as far as we could hope to go with RNS arithmetic for adaptive beamforming may be revised to suggest that $N = 16$ may be amenable to this method of solution.

We should also observe that with even a moderate degree of parallelism available the Gauss-Jordan algorithm becomes a reasonable alternative to Gauss elimination. Since the covariance matrix in our beamforming problem is positive definite Hermitian there is no loss of numerical stability involved. This would allow the elimination above the diagonal to be performed simultaneously with the forward elimination phase and the back substitution would then consist of just one parallel "division" in for which the reciprocation has already been performed. The additional serial operation count of the Gauss-Jordan procedure is of course eliminated in this setting.

7. Conclusions

Clearly time-penalties even close to those suggested by the analysis of Section 2, combined with the increased numbers of divisions which are required for integer-arithmetic Gauss elimination for a positive definite Hermitian system are likely to prove unacceptable for "real-time" computation using the CRT to perform the divisions.

This suggests that for RNS arithmetic to be useful in solving the covariance matrix form of the adaptive beamforming problem either the divisionless form of the algorithm [6] or some form of scaling or alternative division algorithm must be used to avoid or at least reduce the number or cost of the divisions that are needed. Scaling using the L-CRT is seen to offer some reasonable savings such that at least small-scale problems are amenable to this solution technique.

The improved version of the Hitz and Kaltofen division algorithm in conjunction with some minimal pipelining and, perhaps most importantly, the observation that only N reciprocals are needed for the $O(N^3)$ divisions, renders larger scale problems suitable for solution by Gauss elimination using RNS arithmetic.

References

[1] J.L.Buchanan & P.R.Turner, *Numerical Methods and Analysis*, McGraw-Hill, 1992

[2] R.T.Compton, *Adaptive Antennas: Concepts and Performance*, Prentice Hall, 1988

[3] R.T.Gregory & D.W.Matula, *Base conversion in Residue Number Systems*, 3rd Symposium on Computer Aritmetic, IEEE, Washington DC, 1975, pp 117-125.

[4] M.Griffin, M.Sousa & F.J.Taylor, *Efficient scaling in the residue number system*, Proc IEEE Conf on Acoustics, Speech and Signal Processing, IEEE, New York, 1989

[5] M.A.Hitz & E.Kaltofen, *Integer division in residue number systems*, Comp. Sci Tech Report # 93-9, Rensselaer Polytechnic Institute, May 1994

[6] B.J.Kirsch & P.R.Turner, *Modified Gaussian Elimination for Adaptive Beamforming using RNS Arithmetic*, NAWC-AD Tech Report, 94112-50, 1994.

[7] B.J.Kirsch & P.R.Turner, *Adaptive Beamforming using RNS Arithmetic*, Proc ARITH11, IEEE Computer Society, Washington, DC, 1993, pp 36-43.

[8] R.A.Monzingo & T.W.Miller, *Introduction to Adaptive Arrays*, Wiley-Interscience, 1980.

[9] R.Nitzberg, *Effects of errors in adaptive weights*, IEEE Trans AES 12 (1976) 369-373.

[10] P.R.Turner & B.J.Kirsch, *An Analysis of Gauss Elimination for Adaptive Beamforming*, NAWC-AD Tech Report, 1994.

[11] P.R.Turner, *An Improved RNS Division Algorithm*, NAWC-AD Tech Report, 1994 (to be submitted to IEEE Symposium on Computer Arithmetic, Bath, UK, 1995)

[12] J.H.Wilkinson, *The Algebraic Eigenvalue Problem*, Oxford University Press, 1965.

DISTRIBUTION LIST

No. of Copies

US NAVAL ACADEMY.....10
ANNAPOLIS, MD 21402
(ATTN: MATHEMATICS DEPARTMENT, PETER R. TURNER)

AVIONICS DEPARTMENT.....12
ENGINEERING DIVISION (CODE 4.5.5.1)
NAVAL AIR WARFARE CENTER
AIRCRAFT DIVISION WARMINSTER
P. O. BOX 5152
WARMINSTER, PA 18974-0591)
(10 FOR CODE 4.5.5.1, BARRY J. KIRSCH)
(2 FOR CODE 7.2.5.5)

DEFENSE TECHNICAL INFORMATION CENTER.....2
ATTN: DTIC-FDAB
CAMERON STATION BG5
ALEXANDRIA, VA 22304-6145

CENTER FOR NAVAL ANALYSIS.....1
4401 FORT AVENUE
P. O. BOX 16268
ALEXANDRIA, VA 22302-0268

OFFICE OF NAVAL RESEARCH.....2
800 NORTH QUINCY STREET
ARLINGTON, VA 22217-5660
(2 FOR ONR-313)